This listing of claims will replace all prior versions, and listings, of claims in the application:

LISTING OF THE CLAIMS:

1. (Currently Amended) In a computer system using an operating system to provide access to hardware resources, wherein said operating system provides access to said hardware resources via a first ~~source~~ code component, a method of replacing said first ~~source~~ code component with a new ~~source~~ code component while said operating system remains active and while said operating system provides continual availability to ~~applications of~~ the hardware resources by applications operational in the computer system, the method comprising:

identifying references to said first ~~source~~ code component; and

replacing the identified references to said first ~~source~~ code component with references to said new ~~source~~ code component.

2. (Original) A method according to Claim 1, wherein the method is implemented transparently to said applications.

3. (Original) A method according to Claim 1, wherein the method is scalable.

4. (Currently Amended) A method according to Claim 1, wherein said computer system comprises a multiprocessor system, which comprises a plurality of processors, and said method is implemented across each of said processors ~~mechanism is divided across a multiprocessor system so that each processor can proceed~~ independently.

5. (Currently Amended) A method according to Claim 1, wherein the replacing step includes the steps of:

establishing a quiescent state for the first code component;

transferring said identified references at said established quiescent state from the first code component to the new code component; and

after transferring said identified references to the new code segment at said established quiescent state, swapping the first code component with the new code component.

6. (Currently Amended) A method according to Claim 1, wherein the step of identifying references includes the steps of:

separating the first code component into objects; and

grouping said objects into a table, and generating to identify said ~~whereby~~ references to said objects, which identified references are ~~is~~ entered in the table.

7. (Currently Amended) A method according to Claim 1, wherein the replacing step includes the steps of:

establishing a quiescent state for the first code component, without locking the first code component, by tracking active threads to the first code component, and identifying the active threads as said identified references;

transferring said identified references, during the quiescent state, from the first code component to the new code component; and

after transferring said ~~quiescent state~~ identified references, swapping the first code component with the new code component.

8. (Currently Amended) A method according to Claim 1, wherein the replacing step includes the steps of:

establishing a quiescent state for the first code component that includes the identified references;

transferring the identified references at the quiescent state from the first code component to the new code component by providing an infrastructure that operates to negotiate a best transfer algorithm; and

after transferring said identified references at the quiescent state, swapping the first code component with the new code component.

9. (Currently Amended) A method according to Claim 1, wherein:

the step of identifying references includes the steps of

separating the first code component into objects, and

grouping said objects into a table, and generating to identify said ~~whereby~~ references to said objects, which are entered in the table; and

the replacing step includes the steps of

establishing a quiescent state for the first code component that includes said identified references;

transferring said references identified to said objects for ~~the quiescent state from~~ the first code component to the new code component by providing an infrastructure to negotiate a best transfer algorithm; and

after transferring said references to the new code segment ~~quiescent state~~, swapping the first code component with the new code component.

10. (Currently Amended) A system for swapping source code in a computer system including an operating system, said operating system including at least one ~~source~~ code component and providing continual availability of hardware resources by applications operational in the computer system, the system comprising:

means for identifying, while said operating system is active and providing continual access to said resources, references to a first ~~source~~ code component of the operating system; and

means for replacing the first code component with the new code component including first transferring the identified references to the second code component, while said operating system is active and providing continual access to said resources~~, to first source code with references to a new source code component for the operating system~~, and then executing the replacing.

11. (Original) A system according to Claim 10, wherein the system operates transparently to said applications and the system is scalable.

12. (Currently Amended) A system according to Claim 10, wherein said computer system comprises a multiprocessor system, which comprises a plurality of processors, and said method is implemented across each of said processors ~~mechanism is divided across a multiprocessor system so that each processor can process~~ independently.

13. (Currently Amended) A system according to Claim 10, wherein the means for replacing step includes:

means for establishing a quiescent state for the first code component;

means for transferring said references at the established quiescent state from the first code component to the new code component; and

means for swapping the first code component with the new code component after said identified references have ~~quiescent state has~~ been transferred.

14. (Currently Amended) A system according to Claim 10, wherein the means for identifying references includes:

means for separating the first code component into objects; and

means for grouping said objects into a table, and generating to identify said ~~whereby~~ references to said objects, which are entered in the table.

15. (Currently Amended) A system according to Claim 10, wherein the means for replacing includes:

means for establishing a quiescent state for the first code component, without locking the first code component, by tracking active threads to the first code component, and identifying the active threads as said identified references;

means for transferring said identified references, during the quiescent state, from the first code component to the new code component; and

means for swapping the first code component with the new code component after said identified quiescent state references have ~~has~~ been transferred.

16. (Currently Amended) A system according to Claim 10, wherein the means for replacing includes:

means for establishing a quiescent state for the first code component that includes the identified references;

means for transferring the identified references, during the quiescent state, from the first code component to the new code component by providing an infrastructure to negotiate a best transfer algorithm; and

means for swapping the first code component with the new code component after said identified references have ~~quiescent state has~~ been transferred.

17. (Currently Amended) A method according to Claim 10, wherein:

the means for identifying references includes

i) means for separating the first code component into objects, and

ii) means for grouping said objects into a table, and generating to identify said ~~whereby~~ references to said objects, which are entered in the table; and

the means for replacing includes

i) means for establishing a quiescent state for the first code component;

ii) means for transferring the identified references during the quiescent state, from the first code component to the new code component by providing an infrastructure to negotiate a best transfer algorithm; and

iii) means for swapping the first code component with the new code component after said identified references have ~~quiescent state has~~ been transferred.

18. (Currently Amended) A program storage device, for use with a computer system including an operating system to provide access to hardware resources, wherein said operating system provides access to said resources via a first ~~source~~ code component, said program storage device being readable by machine, tangibly embodying a program of instructions executable by the machine to perform method steps for replacing said first ~~source~~ code component with a new ~~source~~ code component while said operating system remains active and while said operating system provides continual availability to ~~applications of~~ said resources by applications operational in the computer system, the method steps comprising:

identifying references to said first ~~source~~ code component; and

replacing the identified references to said first ~~source~~ code component with references to said new ~~source~~ code component.

19. (Currently Amended) A program storage device according to Claim 18, wherein the method is implemented transparently to said applications, the method is scalable, and said computer system comprises a multiprocessor system, which comprises a plurality of processors, and said method is implemented across each of said processors ~~mechanism is divided across a multiprocessor system so that each processor can proceed~~ independently.

20. (Currently Amended) A program storage device according to Claim 18, wherein the replacing step includes the steps of:

establishing a quiescent state for the first code component;

transferring said identified resources at said established quiescent state from the first code component to the new code component; and

after transferring said identified references to the new code segment at said established quiescent state, swapping the first code component with the new code component.

21. (Currently Amended) A program storage device according to Claim 18, wherein the step of identifying references includes the steps of:

separating the first code component into objects; and

grouping said objects into a table, and generating to identify said ~~whereby~~ references to said objects, which identified references are entered in the table.

22. (Currently Amended) A program storage device according to Claim 18, wherein the replacing step includes the steps of:

establishing a quiescent state for the first code component, without locking the first code component, by tracking active threads to the first code component, and identifying the active threads as said identified references;

transferring said identified references, during the quiescent state, from the first code component to the new code component; and

after transferring said identified references~~quiescent state~~, swapping the first code component with the new code component.

23. (Currently Amended) A program storage device according to Claim 18, wherein the replacing step includes the steps of:

establishing a quiescent state for the first code component that includes the identified references;

transferring the identified references at the quiescent state from the first code component to the new code component by providing an infrastructure that operates to negotiate a best transfer algorithm; and

after transferring the identified at said quiescent state, swapping the first code component with the new code component.

24. (Currently Amended) A program storage device according to Claim 18, wherein: the step of identifying references includes the steps of

i) separating the first code component into objects, and

ii) grouping said objects into a table, and generating to identify ~~whereby~~ references to said objects, which are entered in the table; and

the replacing step includes the steps of

i) establishing a quiescent state for the first code component that includes said identified references;

ii) transferring said references identified to said objects for ~~the quiescent state from~~ the first code component to the new code component by providing an infrastructure to negotiate a best transfer algorithm; and

iii) after transferring said identified references ~~quiescent state~~, swapping the first code component with the new code component.